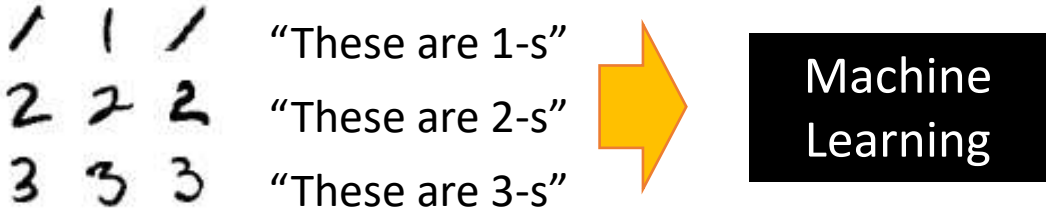
Reinforcement Learning

ITI0210, lecture 14 (2021)

Review

Supervised learning:



show lots of examples

hope the ML model will generalize

In some domains (image/video, 3D sensing, natural language)

this works **well enough**

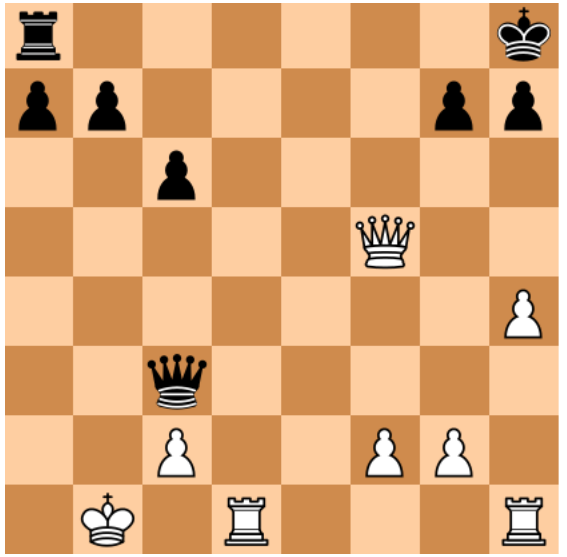
Training Data Crisis

Chess: $3 \cdot 10^6$ high quality grandmaster games to learn from

vs 10^{40} possible positions

Subtle differences between glorious victory/total disaster

Black can force a draw



In chess, supervised learning has always failed

Training Data Crisis

“The AI revolution will not be supervised”

- Alexei Efros, UC Berkeley

Reason: many problems require **unreasonably large** amounts of **labeled** training data

Reinforcement Learning

Learn by Reward and Punishment

Learning to Act

Reinforcement learning:

try something, see if good or bad stuff happens

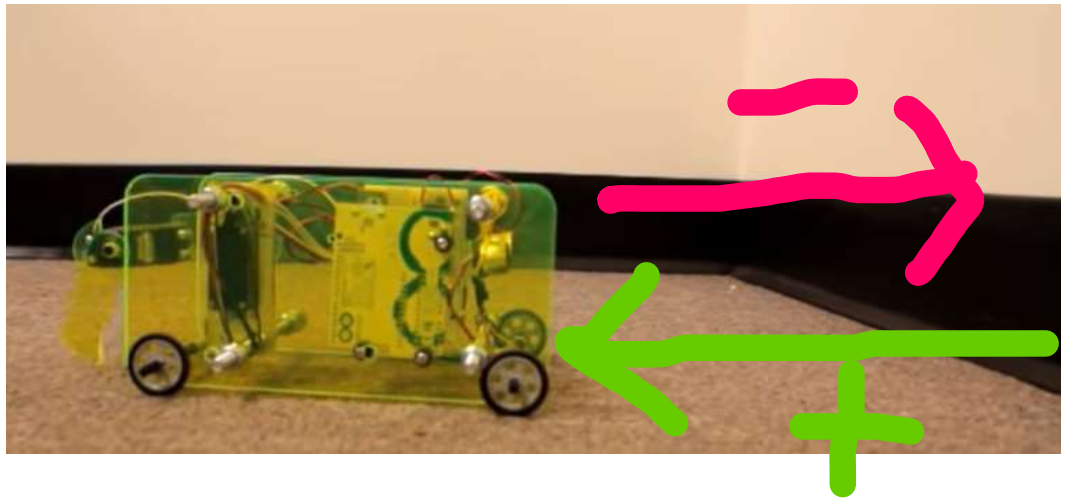
Robot in the video learns to crawl forward:

https://youtu.be/f2nIKFMyfSg?t=335

Q-learning: States, Actions, Rewards

Crawling robot in the video

- 36 states for the arm (6 angles for each joint)
- 4 actions (two servos, two directions)
- reward: distance from wall



Q-learning

Table for choosing actions (with some learned values):

Q(s,a)	a_1	a_2	a_3	a_4
s_1	10	9	-5	0
s_2	0	0	-10	0
…	…	…	…	…
s_{36}	0	0	0	3

In state *s*, choose action *a* with highest Q(*s*, *a*)

Q-learning

Table for choosing actions (with some learned values):

Q(s,a)	a_1	a_2	a_3	a_4
s_1	10	9	-5	0
s_2	0	0	-10	0
…	…	…	…	…
s_{36}	0	0	0	3

Choose random actions sometimes to explore

Q-Learning Algorithm

Q-learning: Updating Q-values

After action a was taken in state s

$$Q(s,a) = Q(s,a) + \alpha\left(\underbrace{r + \gamma \max_{a'} Q(s',a')}_{\text{reward + future gain}} - \underbrace{Q(s,a)}_{\text{previous value}}\right)$$

α controls how fast we learn new stuff / forget old stuff

r – reward from the transition s, a, s'

$\max_{a'} Q(s',a')$ - **best expected** outcome from current state s'

Q-learning: Updating Q-values

After action a was taken in state s

$$Q(s,a) = Q(s,a) + \alpha\left(r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)$$

γ - discount factor

Useful for sequences of actions

Ex: $\gamma = 0.8$; State giving reward 100 is 4 actions away

Current state gets $0.8 \cdot 0.8 \cdot 0.8 \cdot 0.8 \cdot 100 = 40.96$

Q-learning: Updating Q-values

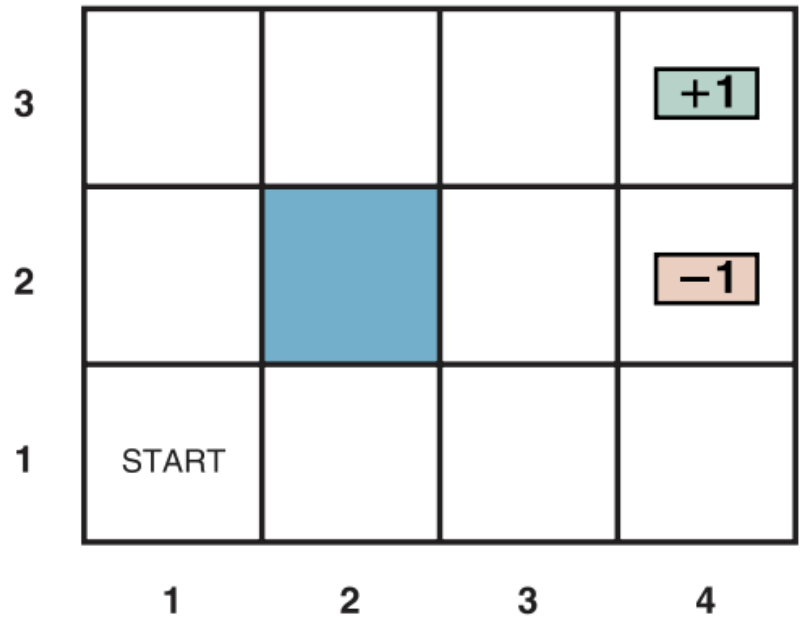
Example:

Learn to walk from start(1,1) to (4,3)

States: squares

Actions: left, right, up, down

Rewards: as pictured (+1, -1)



Initialize Q-table with 0-s, $\alpha = 1, \gamma = 0.8$

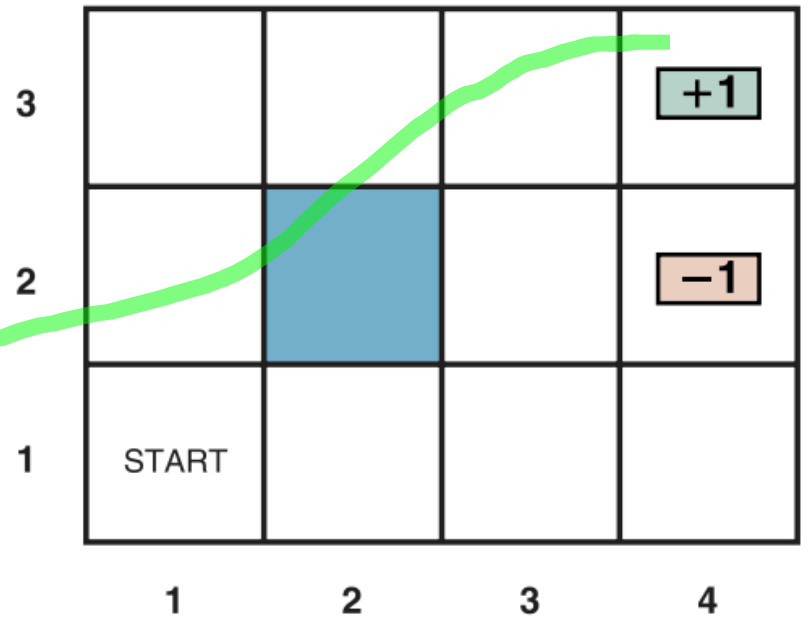
Q-learning: Updating Q-values

Episode 1:

Everything is 0-s until:

pretend we walked to $s = (3,3)$ randomly

then took action $a = right$



$$Q(s,a) = 0 + 1(1 + 0.8 \cdot 0 - 0) = 1$$

Old value

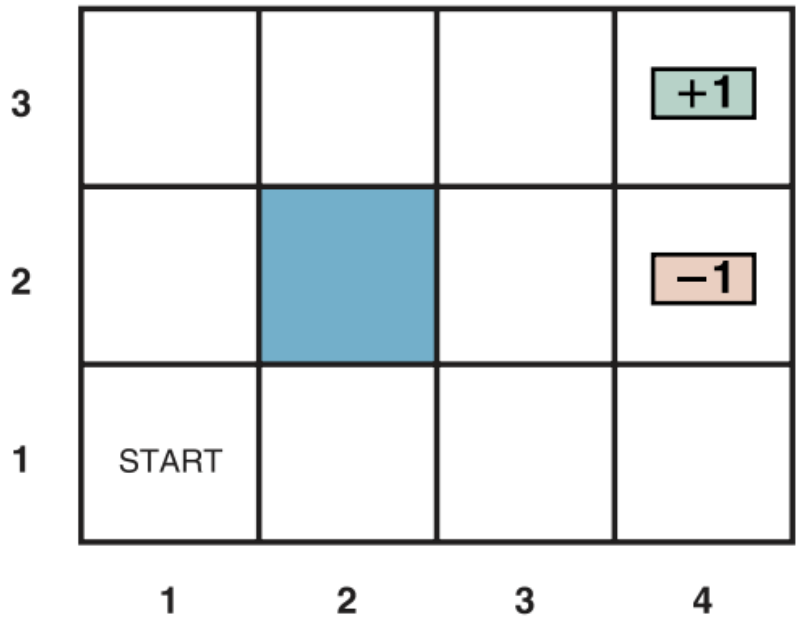
Q-learning: Updating Q-values

Episode 1:

Part of Q-table after

$s = (3,3), a = right$

	up	right	down	left
(4,3)	0	0	0	0
(3,3)	0	1	0	0
(2,3)	0	0	0	0
…	…	…	…	…



Q-learning: Updating Q-values

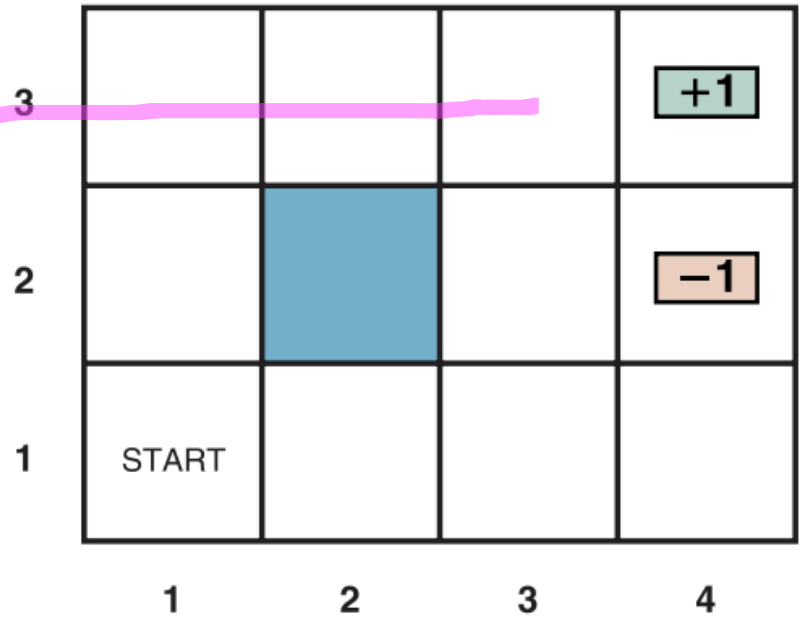
Episode 2:

Pretend we wandered to (2,3)

action $a = right$

$Q(s,a) = 0 + 1(0 + 0.8 \cdot 1 - 0) = 0.8$

	up	right	down	left
(4,3)	0	0	0	0
(3,3)	0	1	0	0
(2,3)	0	0	0	0
…	…	…	…	…

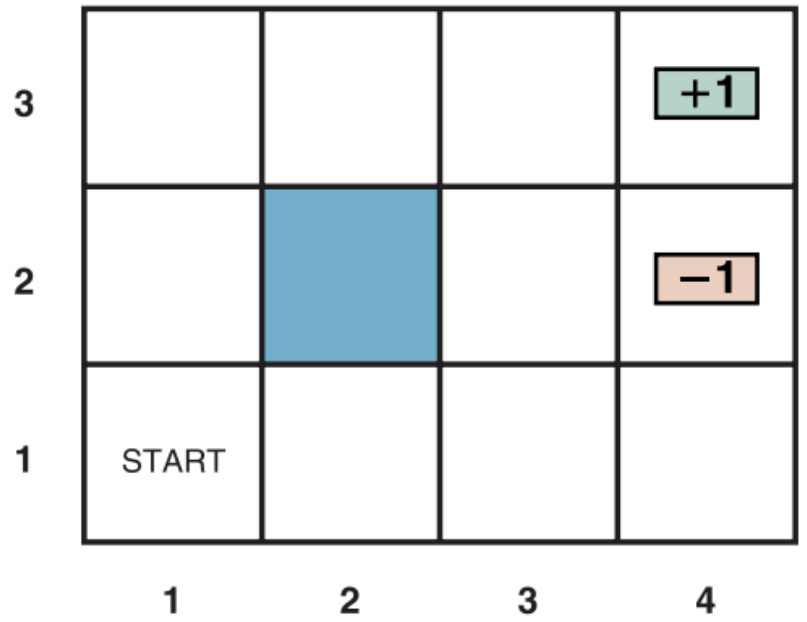


Q-learning: Updating Q-values

We learned the useful path fragment

(2,3) => (3,3) => (4,3)

	up	right	down	left
(4,3)	0	0	0	0
(3,3)	0	1	0	0
(2,3)	0	0.8	0	0
…	…	…	…	…



Exercise: Why is it good that (2,3) has lower value?

Higher values closer to goal make the agent walk the shortes path towards the goal

Choosing Actions

Q-learning, simplified:

```
while episode_not_over:
        a = policy_f(Q, s)
        next_s, r = do_action(a)
        Q[s][a] += alpha * (r + gamma * max(Q[next_s]) - Q[s][a])
        s = next_s
```

Need the policy function $\pi(s)$ to choose the action

- survive long enough to get rewards
- explore the unknown to find better strategies

Bandit Problems

Suppose I believe that:
some slot machines (a.k.a one-armed bandits)
pay out more than others

How much should I **walk around and spend coins** to test machines
vs **use the one performing best so far**

exploration

exploitation

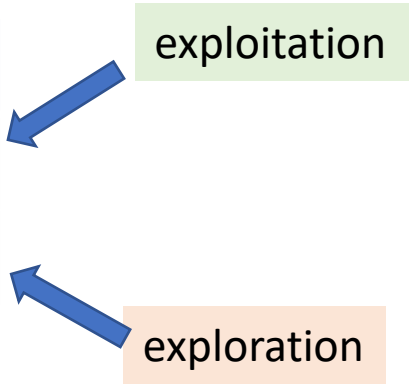
Image credit: Krzysztof Hepner, Unsplash

Choosing Actions

Q-learning also needs to balance exploration and exploitation

ϵ-greedy policy:

Action choice	Probability
a with highest $Q(s,a)$	$1-\epsilon$
uniformly random	ϵ



Common modification:

Start with high ϵ, decrease gradually

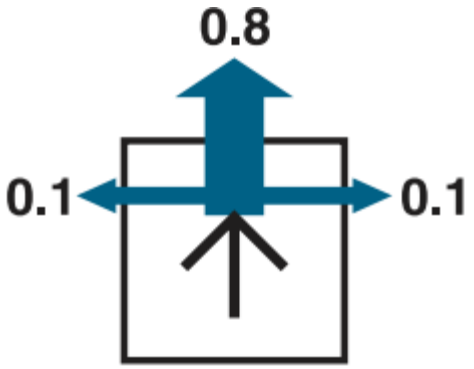
On-policy learning

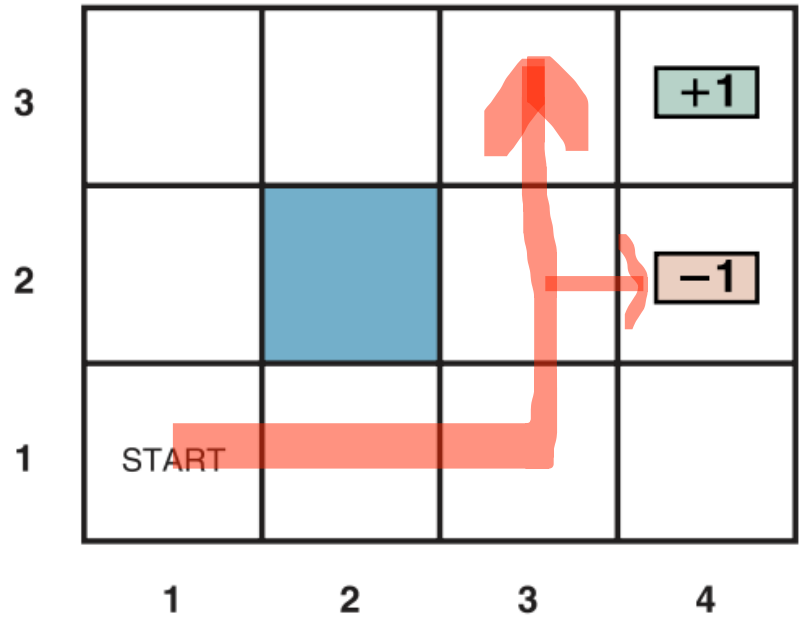
The SARSA algorithm

Being Too Optimistic

Allowing non-determinism (malfunctions, accidents):

Action succeeds with $p = 0.8$, otherwise moves sideways





Dangerous path: at (3,2) there is 10% chance of receiving a penalty (e.g. falling into a pit)

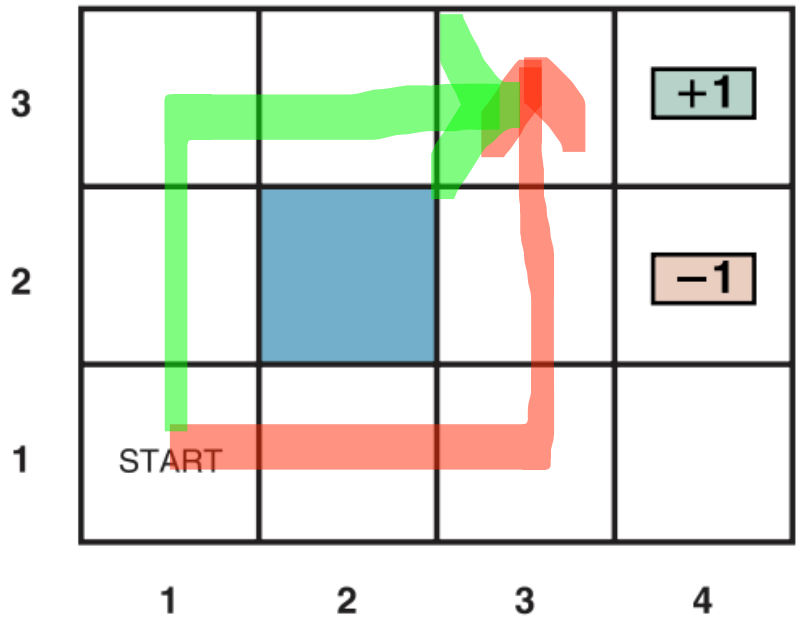
Off Policy vs On-policy

Off policy:

expects best outcome (Q-learning)

On-policy:

see what actually happens (SARSA)



Q-learning cannot distinguish the two paths

because assumes “up” always succeeds from (3,2)

SARSA

After action a' was chosen in state s'

$$Q(s,a) = Q(s,a) + \alpha\big(r + \gamma Q(s',a') - Q(s,a)\big)$$

Q-value of chosen action (could be bad due to exploration policy)

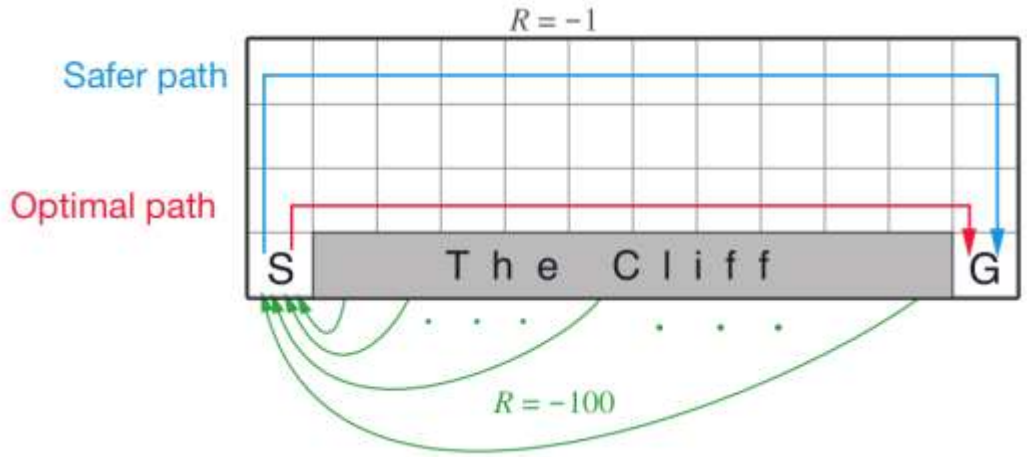
Learning is delayed slightly until a' is known

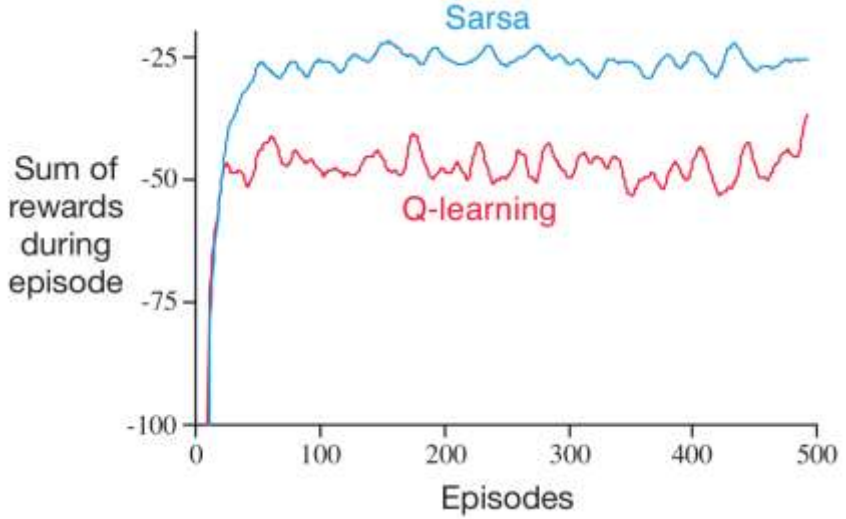
This formula penalizes the **policy**

for choosing the action **after it is known to be bad**

SARSA vs Q-Learning

Cliff walking example where SARSA learns a safer, longer path





This does not mean SARSA dominates Q-learning!

Sutton, Richard S., and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Maze Example